DEPARTMENT OF THE NAVY

OFFICE OF COUNSEL
NAVAL UNDERSEA WARFARE CENTER DIVISION
1176 HOWELL STREET
NEWPORT RI 02841-1708

IN REPLY REFER TO:

Attorney Docket No. 100581
14 Mar 12

The below identified patent application is available for licensing. Requests for information should be addressed to:

TECHNOLOGY PARTNERSHIP ENTERPRISE OFFICE
NAVAL UNDERSEA WARFARE CENTER
1176 HOWELL ST.
CODE 07TP, BLDG. 990
NEWPORT, RI 02841

Serial Number 13/238,372

Filing Date 21 September 2011

Inventor Kevin C. Mattos

Address any questions concerning this matter to the Office of Technology Transfer at (401) 832-1511.

20120319051

DISTRIBUTION STATEMENT
Approved for Public Release
Distribution is unlimited

Attorney Docket No. 100581

SYSTEM AND METHOD FOR A SIMPLE NETWORK MANAGEMENT PROTOCOL DOWNTIME CALCULATOR

STATEMENT OF GOVERNMENT INTEREST

[0001] The invention described herein may be manufactured and used by or for the Government of the United States of America for Governmental purposes without the payment of any royalties thereon or therefor.

CROSS REFERENCE TO OTHER PATENT APPLICATIONS

[0002] None.

BACKGROUND OF THE INVENTION

1) Field of the Invention

[0003] This disclosure relates in general to the field of computer networks and, more particularly, to a system and method for a simple network management protocol (SNMP) downtime calculator.

2) Description of Prior Art

[0004] Suitability of a network system may be evaluated by analyzing reliability, availability and maintainability, (RAM) at a subsystem level to ensure that the subsystems meet specified RAM requirements. In particular, calculation of operational availability (A_o) of systems may involve monitoring downtime, for example, when subsystems break down and are in

need of repair. Monitoring downtime may be performed manually, or automatically, for example, with the use of Simple Network Management Protocol (SNMP) where the SNMP architecture system acts as a system manager, and other systems on the network have SNMP agents reporting SNMP trap events (e.g., downtime events) back to the SNMP architecture system.

[0005] Currently there is a need for evaluation of both RAM and RAM growth through a viable RAM improvement strategy that includes a reliability growth program as an integral part of design and development of large systems. The purpose of the RAM evaluation and growth efforts is to ensure network system quality and longevity. In particular, RAM metrics become increasingly important to quantitatively evaluate quality of software both during development periods and in operational use as complexity of software systems increases. With increased complexity comes an increased level of effort to properly analyze RAM metrics. Implementing a RAM strategy involves collecting data during specific test events and during normal system use.

[0006] However, currently, the ability to evaluate certain systems during normal use in great detail can be difficult. Typical testing periods can produce a large number of SNMP events (e.g., upwards of 100,000), making manual or automatic analysis of downtime and operational availability for each

subsystem potentially impossible without the use of a parsing tool. For example, constant diligence in logging software failures may be required by a work force already occupied with system operation. Automated logs can greatly assist in recording this data, but it can be difficult to analyze these logs to get an accurate picture of the overall subsystem and system RAM on board. In fact, previous efforts to collect RAM data proved that analysis on such data is tedious and difficult and can be error prone.

[0007] For at least these reasons, a system and method to automatically calculate and analyze SNMP downtime from captured failure events of each subsystem in a SNMP network is needed. Embodiments of the present invention can play a large part in helping streamline analysis of RAM and RAM growth, particularly in environments where data collection is much more difficult due to lack of dedicated manual logging of failure data. Embodiments according to the present invention can provide a more accurate picture of how subsystems are performing compared to manual methods. This can aid in reliability growth analysis predictions and ensure the right problems are being fixed.

SUMMARY OF THE INVENTION

[0008] According to an example embodiment of the present invention, a method includes storing a log comprising a

plurality of trap events from corresponding nodes in a subsystem in a network, the plurality of trap events comprising down events and up events, parsing the log to separate down events and up events, pairing down events with corresponding up events, the pairing including matching, for each node in the subsystem, a down event having a severity level with an up event having the same severity level, calculating subsystem downtime, and calculating operational availability of the subsystem. More specific embodiments include determining overlapping downtime and other features.

[0009] According to an example embodiment of the present disclosure, a system includes at least one subsystem in a simple network management protocol (SNMP) network, at least one SNMP manager connected to the network, and a calculator comprising a database operable to store a log of a plurality of trap events from corresponding nodes in the at least one subsystem, the plurality of trap events including down events and up events, a parsing module operable to parse the log to separate down events and up events, a pairing module operable to pair down events with corresponding up events, including by matching, for each node in the subsystem, a down event having a severity level with an up event having a corresponding severity level, a timestamp module operable to identify corresponding times of occurrence of events, and a report module operable to generate a report

comprising down events and corresponding up events with respective times of occurrence. More specific embodiments include the use of a MySQL relational database and other features.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] To provide a more complete understanding of the present disclosure and features and advantages thereof, reference is made to the following description, taken in conjunction with the accompanying FIGURES, wherein like reference numerals represent like parts, in which:

[0011] **FIG. 1** is a simplified block diagram of an embodiment of a system associated with calculating and analyzing SNMP downtime in accordance with the present disclosure;

[0012] **FIG. 2** illustrates example operational steps in a method associated with embodiments of the present disclosure;

[0013] **FIG. 3** shows an example report associated with embodiments of the present disclosure; and

[0014] **FIG. 4** illustrates example operational steps associated with a method according to embodiments of the present disclosure.

DETAILED DESCRIPTION OF THE INVENTION

[0015] It is to be understood that the following disclosure describes several exemplary embodiments for implementing different features, structures, or functions. Exemplary embodiments of components, arrangements, and configurations are described below to simplify the present disclosure. However, these exemplary embodiments are provided merely as examples and are not intended to limit the scope of the disclosure.

[0016] The present disclosure may repeat reference numerals and/or letters in the various exemplary embodiments and across the Figures. provided herein. This repetition is for the purpose of simplicity and clarity and does not in itself dictate a relationship between the various exemplary embodiments and/or configurations discussed in the various Figures.

[0017] Turning to FIG. 1, there is illustrated a simplified block diagram of a system 10 for calculating and analyzing SNMP downtime. Networked subsystems 12 provide SNMP trap messages 14, corresponding to trap events, to a SNMP manager 16. In an example embodiment, SNMP manager 16 is located centrally in the network. SNMP trap messages 14 may contain information regarding the nature of the trap events and message identification (e.g., identification of a node at which the event occurred). SNMP manager 16 can compile the SNMP trap messages 14 to provide SNMP logs 18 corresponding to SNMP trap

event messages 14 to a calculator 20. Subsystems 12, and SNMP manager 16, may be nodes on a network 22. Subsystem 12 may comprise nodes within the subsystem. A node may be any electronic device (e.g., machine device or a mobile device), network element, client, server, peer, service, application, or other object capable of sending, receiving, or forwarding information over communications channels in a network. Calculator 20 can comprise various modules including parsing module 24, pairing module 26, report module 28, time stamp module 30, memory 32, processor 34 and database 36. Processor 34 may comprise database manipulator module 38 and java module 40.

[0018] Embodiments of system 10 according to the present disclosure may provide a means to assess and record system network faults and to calculate downtimes for any number of systems. As used herein, the term "system" encompasses hardware and software systems, for example, systems based on SNMP trap events. Embodiments of the system disclosed herein can operate on data collected from a network featuring SNMP agents (e.g., agents implemented on subsystems 12) that report back to a central system manager (e.g., SNMP manager 16) on the status of their health with conventional SNMP message attributes. Aggregate system downtime as well as downtime associated with

user-selected events of interest can be calculated by calculator 20.

[0019] Embodiments of system 10 according to the present disclosure can calculate individual subsystem downtime by accumulating all node downtime for "down" and "up" events. As used herein, a "down" event encompasses lost communication with a network as reported by a node in network 22 to an SNMP agent residing in corresponding subsystem 12. An "up" event encompasses re-established communication with the network as reported by a node in network 22 to an SNMP agent residing in corresponding subsystem 12. In general a down event may have a corresponding up event associated with a node in subsystem 12.

[0020] In many cases, nodes within a subsystem 12 may report down at the same time, and calculator 20 can run an algorithm that removes any overlapping downtime from the subsystem downtime calculation to present a more accurate assessment of overall individual subsystem downtime. This calculated downtime can be utilized in an availability calculation to present the user with an operational availability (A_o) for each subsystem 12 that reports either an up or down event for a captured testing period (e.g., testing period or time window set by a user).

[0021] For purposes of illustrating certain example techniques of system 10, it is important to understand SNMP networks and operational availability calculations for RAM

determinations. The following foundational information may be viewed as a basis from which the present invention may be properly explained. Such information is offered earnestly for purposes of explanation only and, accordingly, should not be construed in any way to limit the broad scope of the present invention and its potential applications.

[0022] In general, SNMP is an Internet-standard protocol for managing network-attached devices on IP networks. It is used mostly in network management systems to monitor network-attached devices for conditions that warrant administrative attention, such as downtime events. SNMP systems use one or more administrative computers called managers (e.g., SNMP manager 16) to monitor or manage one or more subsystems (e.g., a group of devices, subsystem 12) on a computer network (e.g., network 22). Each managed subsystem executes a software component called an agent which reports information via SNMP to the SNMP manager. An SNMP-managed network can consist of three key components: (1) Managed subsystem; (2) Agent, which is a software that runs on managed subsystems; and (3) Network management system (NMS), which is a software that runs on the SNMP manager.

[0023] As used herein, the term "managed subsystem" encompasses network nodes that implement an SNMP interface allowing unidirectional (read-only) or bidirectional access to node-specific information. Managed subsystems can exchange

node-specific information with NMSs. Managed subsystems can be any type of devices, including, but not limited to, routers, access servers, switches, bridges, hubs, IP telephones, IP video cameras, computer hosts, and printers.

[0024] An agent is a network-management software module that resides on a managed subsystem. An agent has local knowledge of management information and translates that information to or from an SNMP specific form. A network management system (NMS) executes applications that monitor and control managed subsystems. One or more NMSs may exist on any managed network.

[0025] Such SNMP networks (e.g., network 22) may be used to evaluate RAM metrics (e.g., operational availability) in SNMP systems (e.g., submarine systems). Operational availability is defined as the probability that a system will be ready to perform its specified function, in its specified and intended operational environment, when called for at a random point in time. The equation to calculate operational availability (A_o) is provided by the following mathematical relation:

$$A_o = \frac{\text{Up Time}}{(\text{Up Time} + \text{Down Time})} \qquad (1)$$

[0026] In-lab operability tests provide a convenient method for assessing and calculating operational availability for all subsystems involved in testing. Events that bring the system to a down state can be logged and time stamped manually, and once

the system is brought back to an up state, that time stamp can also be logged. A difference between the two time stamps can be calculated, and the downtime for that single event can be found following each test, all downtime for each subsystem is accumulated, and an operational availability can be calculated for each subsystem. However, in situations where system maturity is low, and down events occur simultaneously, calculating downtime from overlapping downtimes can be problematic.

[0027] The following examples illustrate the overlapping downtime issue: assume that subsystem A has a 24 hour test. Down event 1 occurs 1 hour into the test, and is repaired 4 hours into the test (for a total of 3 hours of downtime, 21 hours of uptime). Operational availability (A_o) for subsystem A is:

$$A_o = \frac{21}{(21 + 3)} = 0.875 \qquad (2)$$

[0028] Subsystem B has a 24 hour test. Down event 1 occurs 1 hour into the test, and is repaired 4 hours into the test. Down event 2 occurs 2 hours into the test, and is repaired 6 hours into the test. Downtime for down event 1 is 3 hours; the downtime for down event 2 is 4 hours. But, in this case, the total subsystem downtime is not 3+4 = 7, because there was a period of time when the subsystem was already considered in a

down state, and a second down event occurred. The subsystem cannot be marked down twice, so the time when both events were down may be removed from the total system downtime. The modified Operational Availability calculation is:

$$A_o = \frac{\text{Up Time}}{(\text{Up Time} + (\text{Down Time} - \text{Overlapping downtime}))} \quad (3)$$

where Up time is calculated as Total System Time - (Downtime - Overlapping downtime), in this case 24 - (7-2) = 19. In the example, A_o of Subsystem B is:

$$A_o = \frac{19}{(19 + (7-2))} = 0.791 \quad (4)$$

[0029] The process to manually go through events for each subsystem and remove overlapping downtime is both tedious and error prone. The results can be long lead times for test reports, and inaccurate subsystem A_o numbers due to miscalculations. In particular, during at-sea operations, manually collecting failure data consistently can be a near impossibility. It may not be possible to carry sufficient manpower on board to log such data due to space limitations and budget constraints.

[0030] Automation of data logging may be implemented through a SNMP network where a designated system acts as a system manager, and other systems on the network have SNMP agents reporting SNMP events back to the designated system. These

events are captured and stored in the designated system. They are exportable from the designated system in the form of simple Structured Query Language (SQL) statements. These SNMP SQL statements may be analyzed for their ability to provide necessary data to calculate A_0 for subsystems reporting back to the designated subsystem.

[0031] Although automatic logging of up and down events via a SNMP network is useful, analyzing the logs may have to be performed manually with consequent errors and difficulties. A system for calculating and analyzing SNMP downtime outlined by FIG. 1, can resolve many of these issues. Embodiments of the present disclosure can reduce labor hours, contribute to a more timely production of test reports, and reduce errors by removing manual evaluation of subsystem events to remove overlapping downtime.

[0032] Turning to the infrastructure of FIG. 1, in an example embodiment, calculator 20 may be located on SNMP manager 16 and configured to run analysis on SNMP logs 18 in real time. According to another example embodiment, calculator 20 may be located on a separate network or system that may be connected to SNMP manager 16 via standard network connection or one way point to point interface connection. Such network connection may include wired or wireless connections including Bluetooth, Zigbee, IEEE 802.11x, WiFi Direct, 60 GHz, ultrawideband (UWB),

a USB cable, an HDMI cable, etc. Such an architecture may be used for post test analysis. According to yet another example embodiment, which may be suitable for post test analysis, calculator 20 may not be connected to SNMP manager 16, and SNMP logs 18 may be passed via alternate means such as removable media (e.g., sneaker-net through magnetic tape, floppy disks, compact discs, CD archives, USB flash drives, external hard drives, etc.).

[0033] According to embodiments of the present disclosure, database 36 in calculator 20 can be a standard relational database, for example, MySQL database that runs as a server providing multi-user access to database 36. SNMP trap events may be stored in database 36 and may be parsed by parsing module 24 to determine down and up events. The down and up events may be configurable within database 36. Therefore, the notion of paired events can be suited to specific SNMP trap events if such events exist.

[0034] Pairing module 26 may pair down and up events corresponding to the same node. In an example embodiment, a default configuration may be to key on standard node down-node up events. Each of these events may be time-stamped with a standard UNIX time stamp when they are reported to SNMP manager 16, or they may be time stamped by time stamp module 30 in calculator 20. Report module 28 may present a user with a

report containing paired down and up events for nodes in corresponding subsystems 12. Mapping of nodes to subsystems and corresponding severities are also configurable within database 36.

[0035] Database manipulator 38 in processor 36 may save SNMP logs 18 as text files. Database manipulator 38 may communicate with database 36, java module 40, and memory 32 to execute a method (e.g., an algorithm) for calculating downtime for individual subsystems. Processor 32 may send a list of complete down-up events for every node in network 22 for a given time period based on standard SNMP messages captured by SNMP manager 16 to report module 28.

[0036] In an example embodiment according to the present disclosure, system 10 may be implemented in a software entity that can easily be transported to a laptop or desktop free of the environment it is analyzing. Software may be written in Java, and therefore can run under Java Virtual Machine (JVM) in any operating system that supports the JVM (e.g., tested under Windows 2000, Windows XP, and Red Hat Linux 8.1). The software can interface with database 36 that contains SNMP trap events. As the trap events are exported as SQL statements, they may be brought into database 36 using simple, known methods.

[0037] Turning to FIG. 2 there is illustrated an example of operational steps in method 50 according to embodiments of the

present disclosure. Method 50 starts in step 52 when SNMP logs 18 are retrieved by calculator 20. Database manipulator 38 may save SNMP logs 18 as text files containing SQL script from tests on subsystems 12 in step 54. In step 56, SQL scripts may be run through a compiler (e.g., MySQL compiler), and the processed logs may be stored in database 36 sorted by down date. Java module 40 in processor 34 may convert UNIX time (if any) into real date stored in database 36 in step 58.

[0038] A database status may give a user the date range of all data in database 36 in standard mm/dd/yyyy format (converted from Unix time).. A starting and ending date and time for a specific analysis may be entered. In an example embodiment, the user may be prompted to enter the starting date and time period that the user wishes to analyze, followed by an ending date and time. The user may also put a limit on the severity of the events analyzed. The user may run parsing module 24.

[0039] In step 60, faulted and failed events are pulled out and corresponding up events are found by Java module 40 in conjunction with parsing module 24 and pairing module 26. Report module 28 shows (e.g., in table format) a list of all singular events (paired up and down events) with their corresponding subsystem and a mm/dd/yyyy time stamp of when the event was reported down, and when it was reported back up. Downtime is calculated in step 62 as the time between down

(e.g., faulted) and up (e.g., cleared) events. In step 64 the calculated downtime is used to determine operational availability. The process ends in step 66.

[0040] Turning to FIG. 3, there is illustrated an example report 70 of down and corresponding up events according to embodiments of the present disclosure. Report module 28 may present a user with a table of single events (e.g., consisting of both the down and up messages) and the time they occurred. For example, column 72 may present a title of the captured event. Each node, which is associated with a corresponding subsystem 12 that owns the node, is also presented to the user for each event, for example, in column 74. Column 76 may display subsystem 12 corresponding to the node. Columns 78 and 80 may display the down and up times, respectively, corresponding to the event. Downtime for each event can be presented to the user as well, for example, in column 82. The report shown in FIG. 3 is illustrated as an example, and not a limitation of the present disclosure. Various other formats and presentation modes not disclosed herein may be used to display the information. The table may be presented on a computer monitor or other suitable user interface. Alternatively and additionally, the user may generate a printout of the report.

[0041] Turning to FIG. 4, there is illustrated an example of operational steps associated with a method 100 according to

embodiments of the present disclosure. In an example embodiment, method 100 may be used for matching down events with up events and it may be based on a matching of event severities for messages with the same ID (e.g., for a down event and corresponding up event) obtained from a SNMP trap message 14. Calculator 20 can provide for a subsystem level A_o analysis. This analysis includes method 100 to remove overlapping downtime from subsystem A_o calculations. Method 100 may be detailed in the table below:

```
Sort SNMP event database by reporting time in ascending order
Set desired severity levels to indicate up and down events:
sevLevelUp, sevLevelDown
Set desired time window:
timeStart, timeEnd
Set all subsystem downtimes to 0
For each event e in database where timestamp of e (tsE) >=timeStart AND <=timeEnd
If e.sevLevel = sevLevelDown
{
        eID = e.ID
        While not done parse each event f in database starting at e+1
        If f.sevLevel = sevLevelUp AND f.ID = eID
        {
                If timestamp of f (tsF) > tsE
                {
                        eventDowntime=tsF-tsE
                        done = true
                }
        }
        Get parent subsystem for node, subsystem
        If subsystem has no previous errors
                Subsystem.eventDowntime = eventDowntime
                set subsystem.downAt = tsE
                set subsystem. upAt = tsF
        else
                if tsE <= subsystem.upAt
                {
```

```
                If(tsF <= subsystem.UpAt)
                Do nothing; time overlaps
                Else if (tsF-subsystem.UpAt>0)
                {
                        slightOverlap = tsF-subsystem.UpAt
                        subsystem.downtime += slightOverlap
                        subsystem.upAt = tsF
                }
                Else if(tsF – subsystem.DownAt>0)
                {
                        No overlaps
                        Subsystem.downAt = tsE
                        Subsystem.upAt = tsF
                        Subsystem.downtime += eventDowntime
                }
        }
        Else
        {
                No overlaps
                Subsystem.downAt = tsE
                Subsystem.upAt = tsF
                Subsystem.downtime += eventDowntime
        }

}
```

[0042] As illustrated in FIG. 4, method 100 starts in step 102 when calculator 20 is ready to perform A_o analysis, for example, when commanded by a user. In step 104, SNMP trap events in database 36 may be sorted in ascending order. In step 106, severities may be assigned to up and down events. In an example embodiment, severity levels captured from the SNMP trap events may be categorized into a range from 0 for informational events to 5 for system failure. In an example embodiment, the user may assign severities. Alternatively, severities may be

pre-assigned by calculator 20 based on the nature of the SNMP event.

[0043] In step 108, desired time window for analysis may be set. In an example embodiment, the user may be prompted to enter the starting date and time period that the user wishes to analyze, followed by an ending date and time. In step 110, message number E is assigned a value of 1 (i.e., first message in the database corresponding to an event in the time window). The event corresponding to the message (i.e., event (E)) is parsed in step 112. A timestamp of the event is obtained and set to variable tsE in step 114. If event (E) is not within the desired time window (set earlier in step 108), variable E is advanced by 1 in step 118. Database 36 may be checked in step 120 to determine if all events have been analyzed. If all events have been analyzed, the process ends in step 122. Otherwise, the next event in database 36 is parsed according to step 112.

[0044] If event is within time window as determined in step 116, event severity is checked to determine if it matches down severity in step 124 (i.e., event(E) corresponds to a down event). If not, the processing proceeds to the next event as indicated in step 118. Otherwise, if severity of event (E) matches down severity, event identification number (ID) (e.g., corresponding to message identification number) is obtained and variable EID is set to the event ID in step 126.

[0045] The calculations proceed to determine downtime for all events occurring after event corresponding to message E. In step 128, message number variable F is set to E+1. Event corresponding to message F (i.e., event (F)) is parsed in step 130. In step 132, event ID (e.g., corresponding to message ID) is obtained and variable FID is set to event ID. If FID is not the same as EID (obtained in step 126) as determined in step 134, database 36 is checked in step 136 to determine if events are over. If the events are not over, the calculation proceeds to the next event, and message number variable F is advanced by 1 in step 138. If all events are over as determined in step 136, the calculation proceeds to the next event in the time window corresponding to message number E+1 in step 118.

[0046] Next, down events are paired with corresponding up events for the same ID (e.g., node), the pairing comprising matching, for each node in the subsystem, a down event having a severity level with an up event having a designated corresponding severity level. If FID is the same as EID as determined in step 134, event severity is checked to determine if it matches up-severity in step 140 (i.e., event (F) is an up event with same severity as event (E)). If FID is not equal to EID (i.e., event (F) does not correspond to the same node as event (E)), the calculation loops back to step 136 to determine if events in database 36 are over. If event severity matches

up-severity, event timestamp is obtained in step 142 and timestamp value is assigned to variable tsF. If tsF (obtained in step 142) is not greater than tsE (obtained in step 114), (i.e., event (F) occurred after event (E)) as determined in step 144, the calculation loops back to step 136 to determine if events in database 36 are over. Otherwise, event downtime is set to the difference between tsF and tsE (i.e., EVENT DOWNTIME = tsF-tsE) in step 146. Parent subsystem 12 (e.g., corresponding to node at which event occurred) for the node/event is obtained in step 148.

[0047] In step 150, a determination of any previous errors in subsystem 12 is made. If previous errors do not exist, subsystem event downtime is advanced by the previously calculated event downtime (calculated in step 146) in step 152. A subsystem down-at-time (SUBSYSTEM.DownAt) is identified corresponding to a time of occurrence of a last previous down event whose event downtime has been calculated. A subsystem up-at-time (SUBSYSTEM.UpAt) corresponding to a time of occurrence of the corresponding up event is also identified. SUBSYSTEM.DownAt is updated to tsE and SUBSYSTEM.UpAt is updated to tsF also in step 152.

[0048] If previous errors exist, tsE is compared with SUBSYSTEM.UpAt in step 154. If tsE is greater than SUBSYSTEM.UpAt (i.e., current down event occurred after the last

up event in the subsystem), there is no overlapping downtime, and subsystem event downtime is advanced by the previously calculated event downtime in step 152. On the other hand, if tsE is less than or equal to SUBSYSTEM.UpAt (i.e., current down event occurred before the last up event in the subsystem), tsF is compared to SUBSYSTEM.UpAt in step 156. If tsF is less than or equal to SUBSYSTEM.UpAt (i.e., current up event occurred before the last up event in the subsystem), there is an overlap in downtime requiring no update to the subsystem event downtime as shown in step 162.

[0049] If tsF is greater than SUBSYSTEM.UpAt, tsF-SUBSYSTEM.UpAt is compared to 0 in step 158. If tsF-SUBSYSTEM.UpAt is greater than 0 (i.e., current up event occurred after the last up event in the subsystem), it implies a slight overlap over SUBSYSTEM.UpAt (i.e., SLIGHT OVERLAP = (tsF-SUBSYSTEM.UpAt)) as indicated in step 160. Subsystem event downtime is updated to add the slight overlap corresponding to the difference between the second timestamp and the subsystem up-at-time (i.e., SUBSYSTEM.DOWNTIME + = SLIGHT OVERLAP) and SUBSYSTEM.UpAt is updated to tsF.

[0050] If tsF-SUBSYSTEM.UpAt is not greater than 0 (i.e., current up event occurred before the last up event in the subsystem), tsF-SUBSYSTEM.DownAt is compared to 0 in step 159. If tsF-SUBSYSTEM.DownAt is greater than 0, there is no

overlapping downtime, and subsystem event downtime is advanced by the previously calculated event downtime in step 152. If tsF-SUBSYSTEM.DownAt is not greater than 0, then subsystem event downtime requires no update to the subsystem event downtime as shown in step 162. A subsystem down-at-time (SUBSYSTEM.DownAt) is identified corresponding to a time of occurrence of a last previous down event whose event downtime has been calculated. A subsystem up-at-time (SUBSYSTEM.UpAt) corresponding to a time of occurrence of the corresponding up event is also identified. SUBSYSTEM.DownAt is updated to tsE and SUBSYSTEM.UpAt is updated to tsF also in step 152. Thus, overlapping downtimes can be eliminated accurately from calculation of subsystem event downtime.

[0051] In example embodiments, the operations as outlined herein may be implemented by logic encoded in one or more tangible media, which may be inclusive of non-transitory media (e.g., embedded logic provided in an ASIC, digital signal processor (DSP) instructions, software potentially inclusive of object code and source code to be executed by a processor or other similar machine, etc.). In some of these instances, one or more memory elements (e.g., memory element 32) can store data used for the operations described herein. This includes the memory elements being able to store software, logic, code, or

processor instructions that are executed to carry out the activities described in this Specification.

[0052] Additionally, calculator 20 and associated or integrated components may include processing elements (e.g., processor 34, etc.) that can execute software or algorithms to perform activities to enable calculating and analyzing SNMP downtime, and to route packets using suitable routing protocols. A processor can execute any type of instructions associated with the data to achieve the operations detailed herein in this Specification. In one example, the processors (as shown in various FIGS.) could transform an element or an article (e.g., data) from one state or thing to another state or thing. In another example, the activities outlined herein may be implemented with fixed logic or programmable logic (e.g., software/computer instructions executed by a processor) and the elements identified herein could be some type of a programmable processor, programmable digital logic (e.g., an FPGA, an EPROM, an EEPROM), or an ASIC that includes digital logic, software, code, electronic instructions, flash memory, optical disks, CD-ROMs, DVD ROMs, magnetic or optical cards, other types of machine-readable mediums suitable for storing electronic instructions, or any suitable combination thereof. Any of the potential processing elements, modules, microprocessors, digital signal processors (DSPs), and other devices described in this

Specification should be construed as being encompassed within the broad term 'processor.'

[0053] While certain embodiments in the present disclosure have been described with reference to submarine systems, the embodiments may be also used with other applications and scenarios. For example, embodiments according to the present disclosure may be applied in general to systems implementing SNMP architecture, such as automated production lines, computerized logistics management, plant facility operations, financial networks, railroad networks, etc. Embodiments of the methods (e.g., method 100) for automatically removing overlapping downtime to calculate operational availability for a system can be used for any situation where a network system can have multiple faults occur simultaneously and downtime for each fault is measured independently.

[0054] Note that in this Specification, references to various features (e.g., elements, structures, modules, components, steps, operations, characteristics, etc.) included in "one embodiment", "example embodiment", "an embodiment", "another embodiment", "some embodiments", "various embodiments", "other embodiments", "alternative embodiment", and the like are intended to mean that any such features are included in one or more embodiments of the present disclosure, but may or may not necessarily be combined in the same embodiments.

[0055] It will be understood that many additional changes in the details, materials, steps and arrangement of parts, which have been herein described and illustrated in order to explain the nature of the invention, may be made by those skilled in the art within the principle and scope of the invention as expressed in the appended claims.

Attorney Docket No. 100581

SYSTEM AND METHOD FOR A SIMPLE NETWORK MANAGEMENT PROTOCOL DOWNTIME CALCULATOR

ABSTRACT OF THE DISCLOSURE

A system and method for simple network management protocol downtime calculator includes storing a log comprising a plurality of trap events from corresponding nodes in a subsystem in a network, the plurality of trap events comprising down events and up events, parsing the log to separate down events and up events, pairing down events with corresponding up events, the pairing including matching, for the at least one node in the subsystem, a down event having a severity level with an up event having the corresponding severity level, calculating subsystem downtime, and calculating operational availability of the subsystem. Additional features include identifying overlapping downtimes in the subsystem.

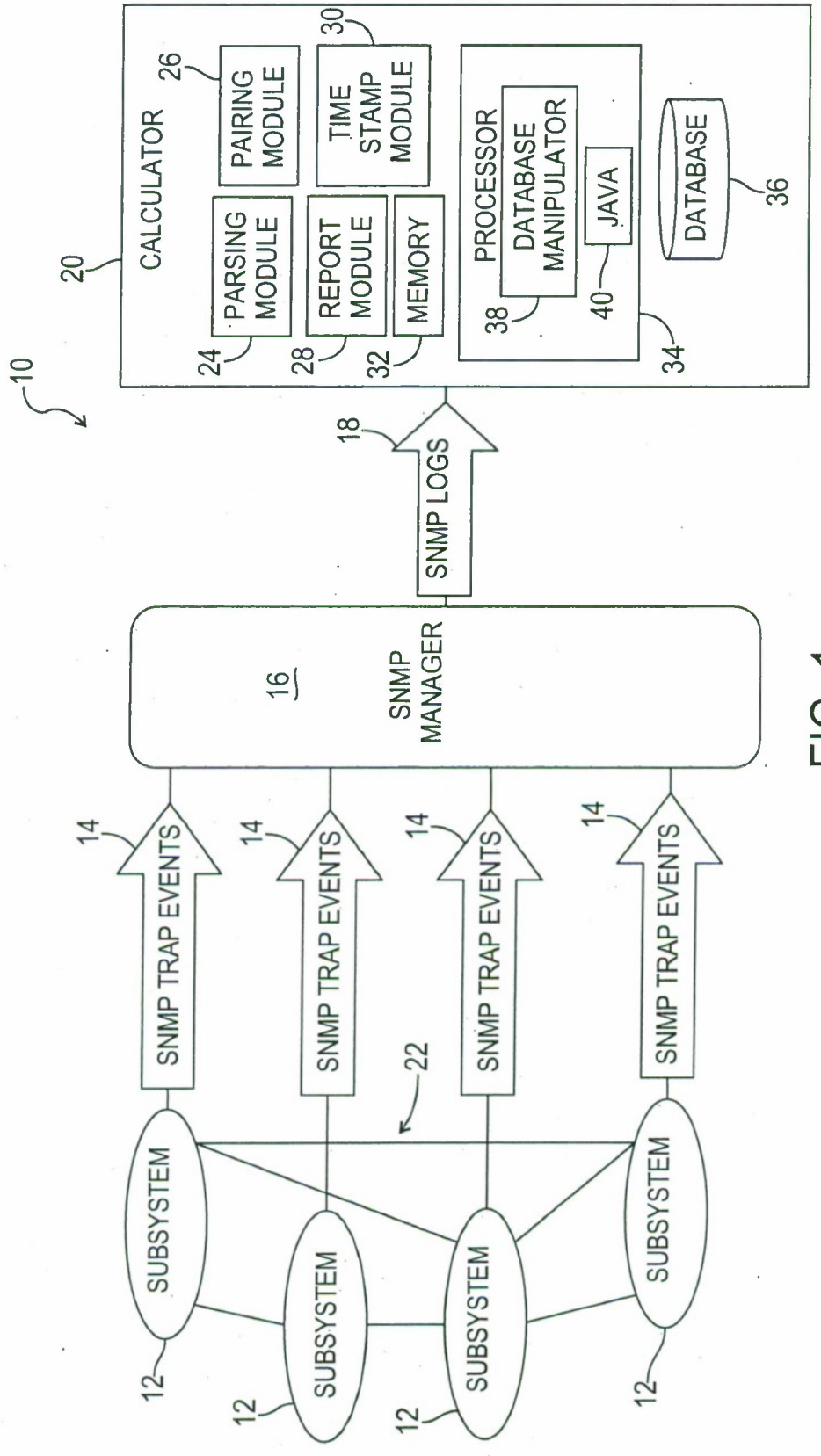
10
20
CALCULATOR
PARSING MODULE
24
PAIRING MODULE
26
REPORT MODULE
28
TIME STAMP MODULE
30
MEMORY
32
PROCESSOR
DATABASE MANIPULATOR
38
JAVA
40
34
DATABASE
36
18
SNMP LOGS
16
SNMP MANAGER
14
SNMP TRAP EVENTS
22
SUBSYSTEM
12

FIG. 1

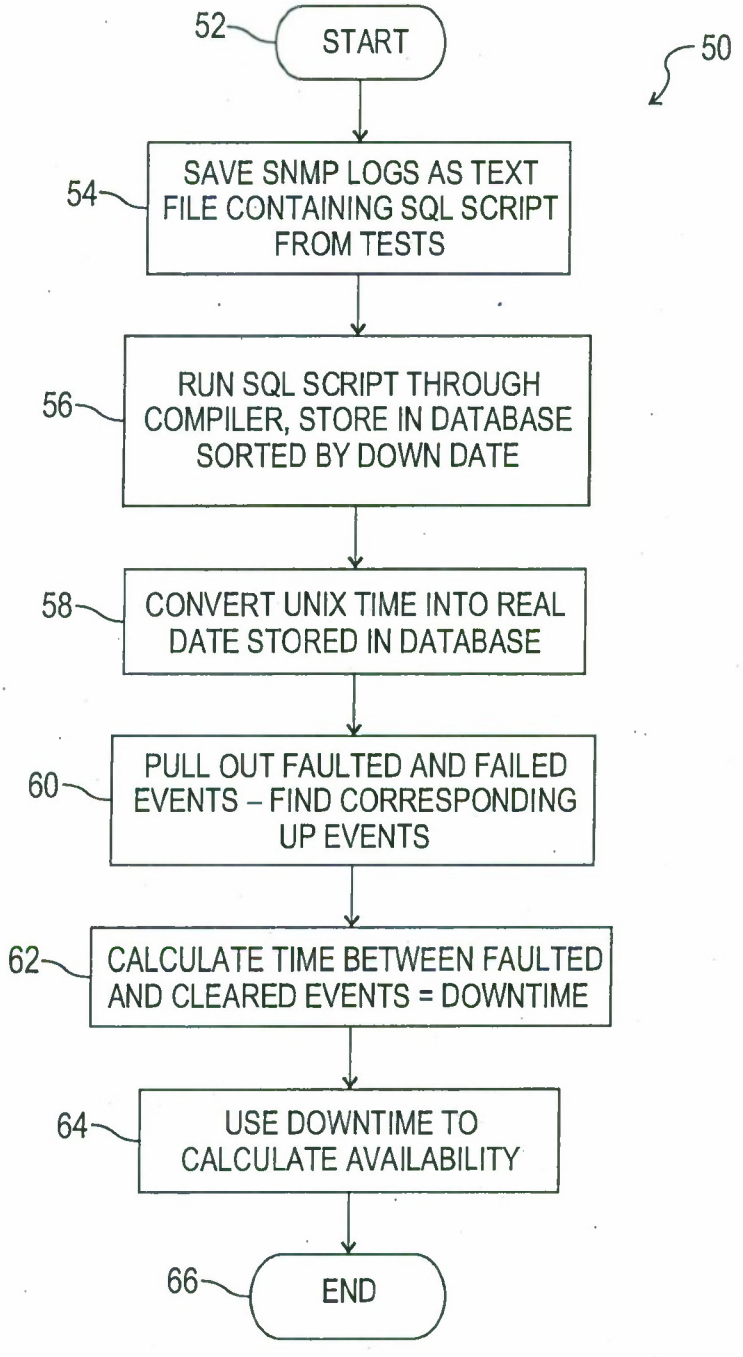
52
START
50
54
SAVE SNMP LOGS AS TEXT FILE CONTAINING SQL SCRIPT FROM TESTS
56
RUN SQL SCRIPT THROUGH COMPILER, STORE IN DATABASE SORTED BY DOWN DATE
58
CONVERT UNIX TIME INTO REAL DATE STORED IN DATABASE
60
PULL OUT FAULTED AND FAILED EVENTS – FIND CORRESPONDING UP EVENTS
62
CALCULATE TIME BETWEEN FAULTED AND CLEARED EVENTS = DOWNTIME
64
USE DOWNTIME TO CALCULATE AVAILABILITY
66
END

FIG. 2

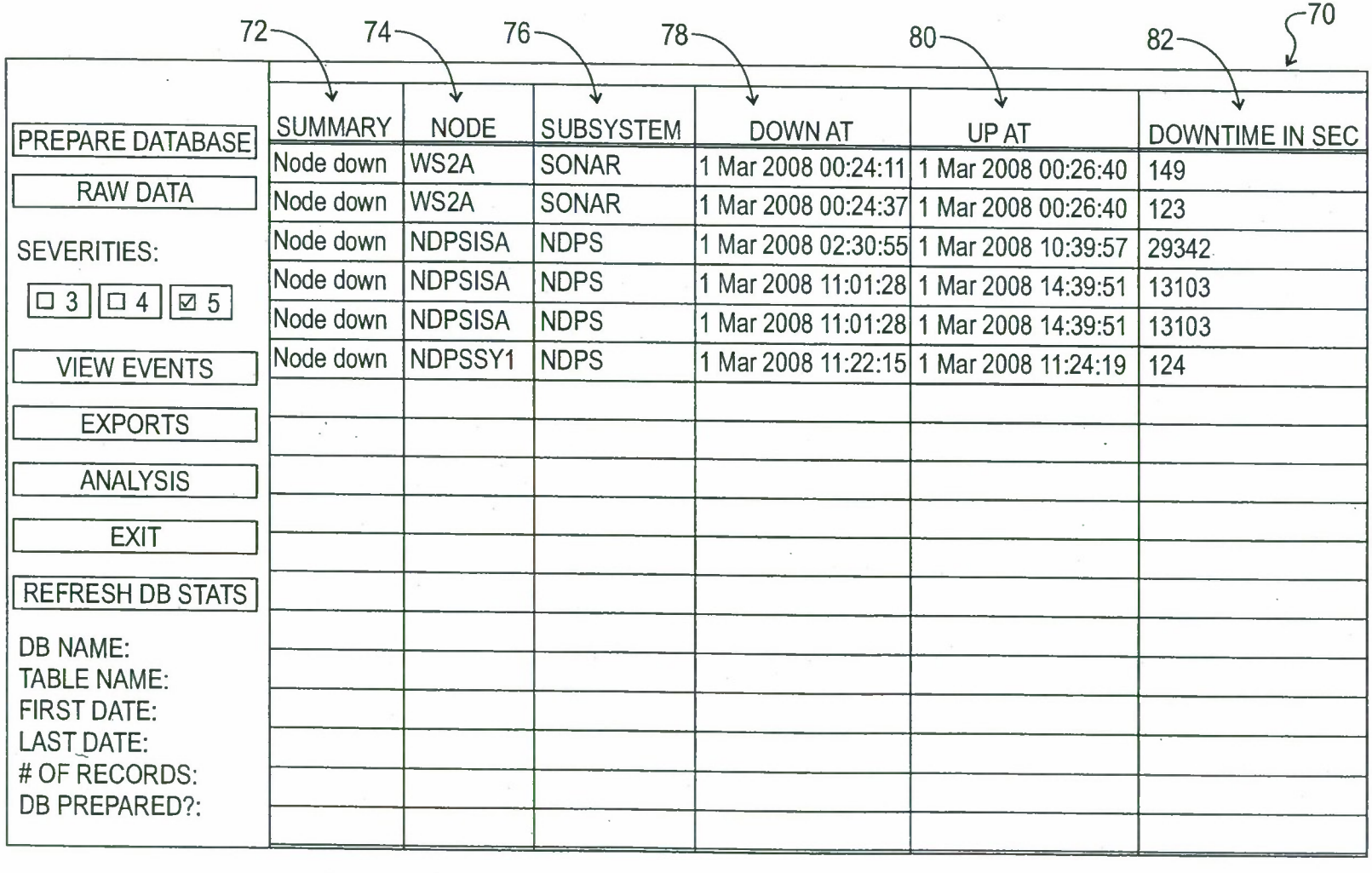
72
74
76
78
80
82
70
PREPARE DATABASE
RAW DATA
SEVERITIES:
☐ 3 ☐ 4 ☑ 5
VIEW EVENTS
EXPORTS
ANALYSIS
EXIT
REFRESH DB STATS
DB NAME:
TABLE NAME:
FIRST DATE:
LAST DATE:
OF RECORDS:
DB PREPARED?:
SUMMARY
NODE
SUBSYSTEM
DOWN AT
UP AT
DOWNTIME IN SEC
Node down WS2A SONAR 1 Mar 2008 00:24:11 1 Mar 2008 00:26:40 149
Node down WS2A SONAR 1 Mar 2008 00:24:37 1 Mar 2008 00:26:40 123
Node down NDPSISA NDPS 1 Mar 2008 02:30:55 1 Mar 2008 10:39:57 29342
Node down NDPSISA NDPS 1 Mar 2008 11:01:28 1 Mar 2008 14:39:51 13103
Node down NDPSISA NDPS 1 Mar 2008 11:01:28 1 Mar 2008 14:39:51 13103
Node down NDPSSY1 NDPS 1 Mar 2008 11:22:15 1 Mar 2008 11:24:19 124

FIG. 3

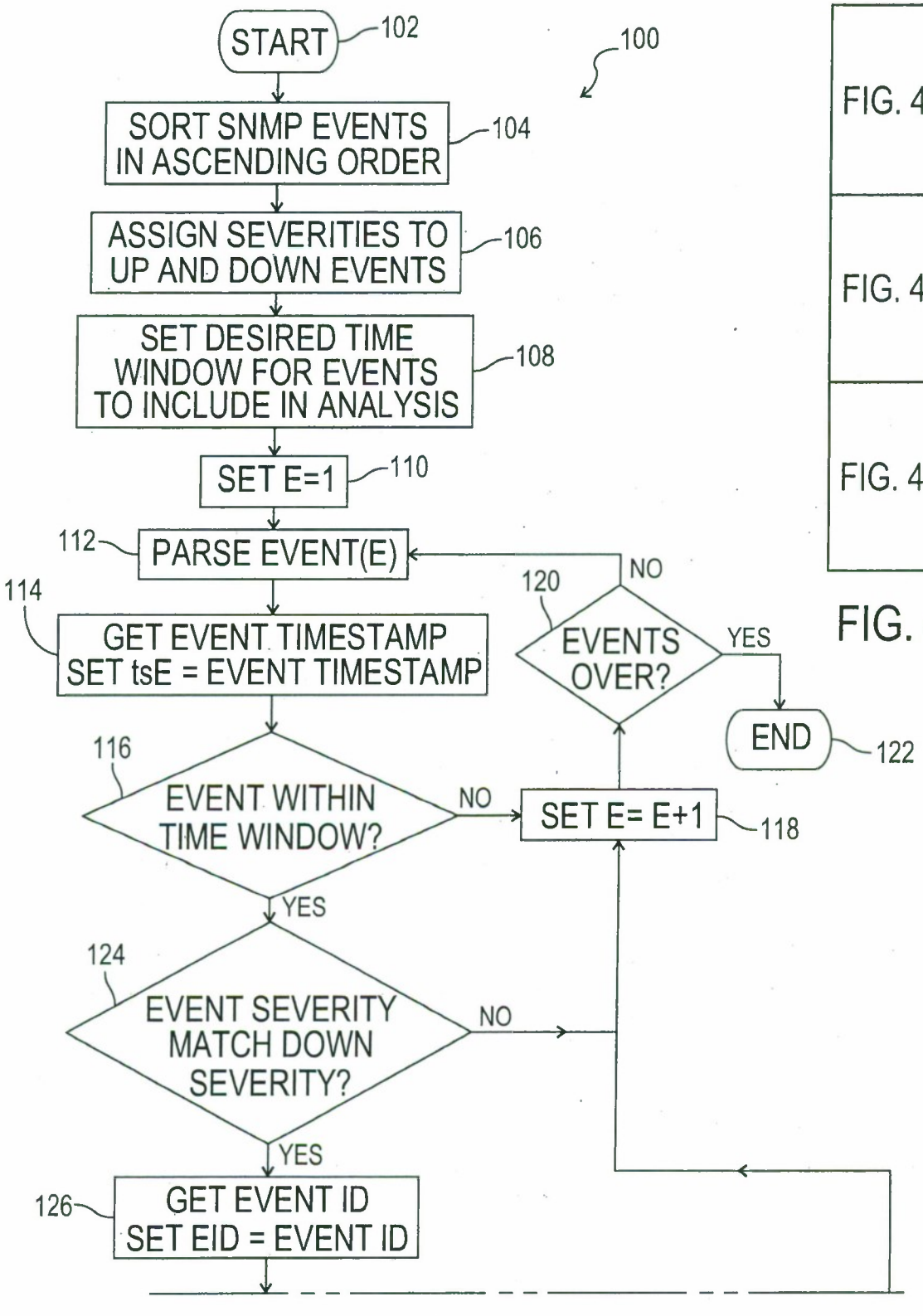

FIG. 4A

FIG. 4B

FIG. 4C

FIG. 4

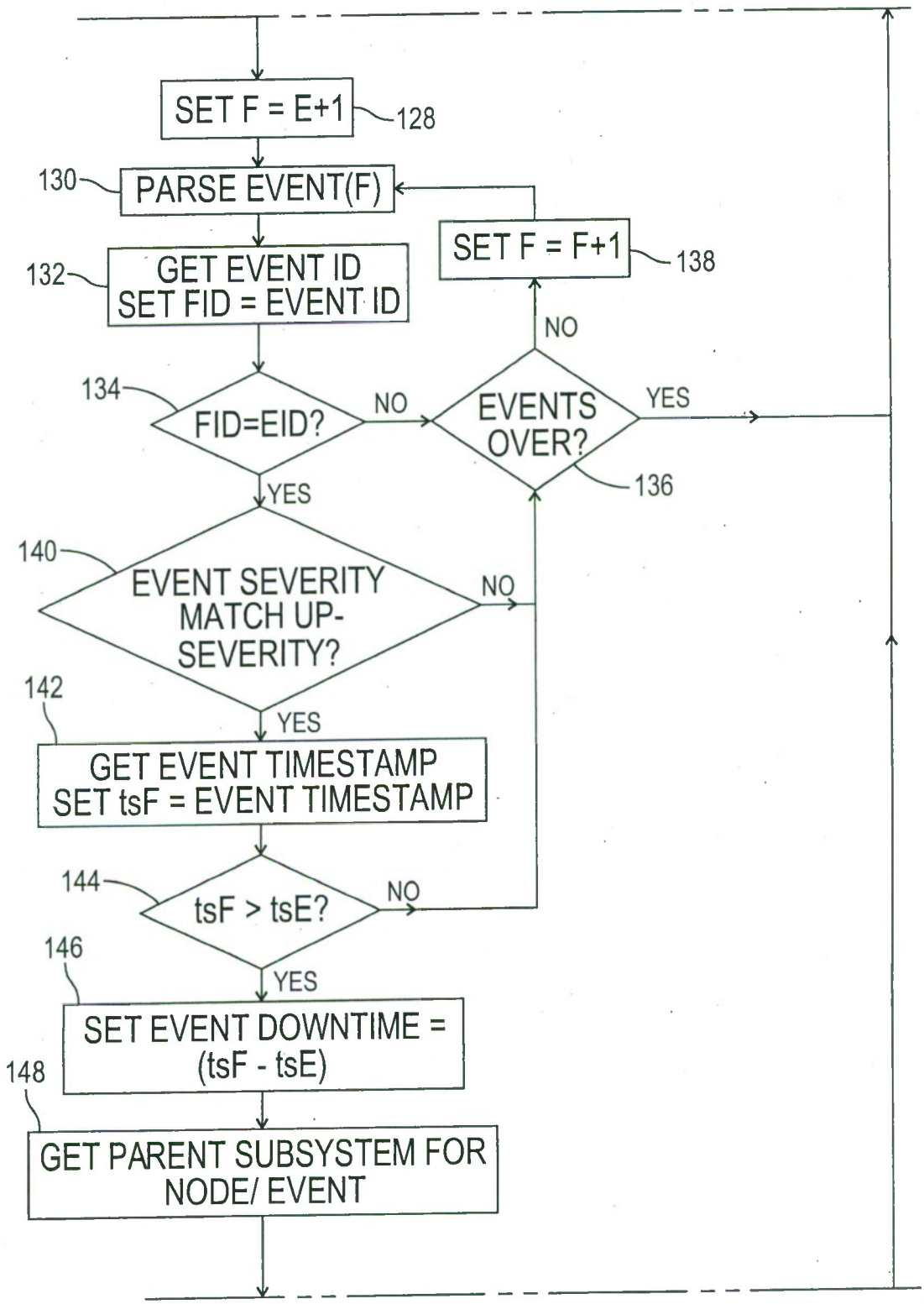
SET F = E+1
128
130
PARSE EVENT(F)
132
GET EVENT ID
SET FID = EVENT ID
134
FID=EID?
NO
EVENTS OVER?
YES
136
NO
SET F = F+1
138
YES
140
EVENT SEVERITY MATCH UP-SEVERITY?
NO
142
YES
GET EVENT TIMESTAMP
SET tsF = EVENT TIMESTAMP
144
tsF > tsE?
NO
146
YES
SET EVENT DOWNTIME =
(tsF - tsE)
148
GET PARENT SUBSYSTEM FOR NODE/ EVENT

FIG. 4B

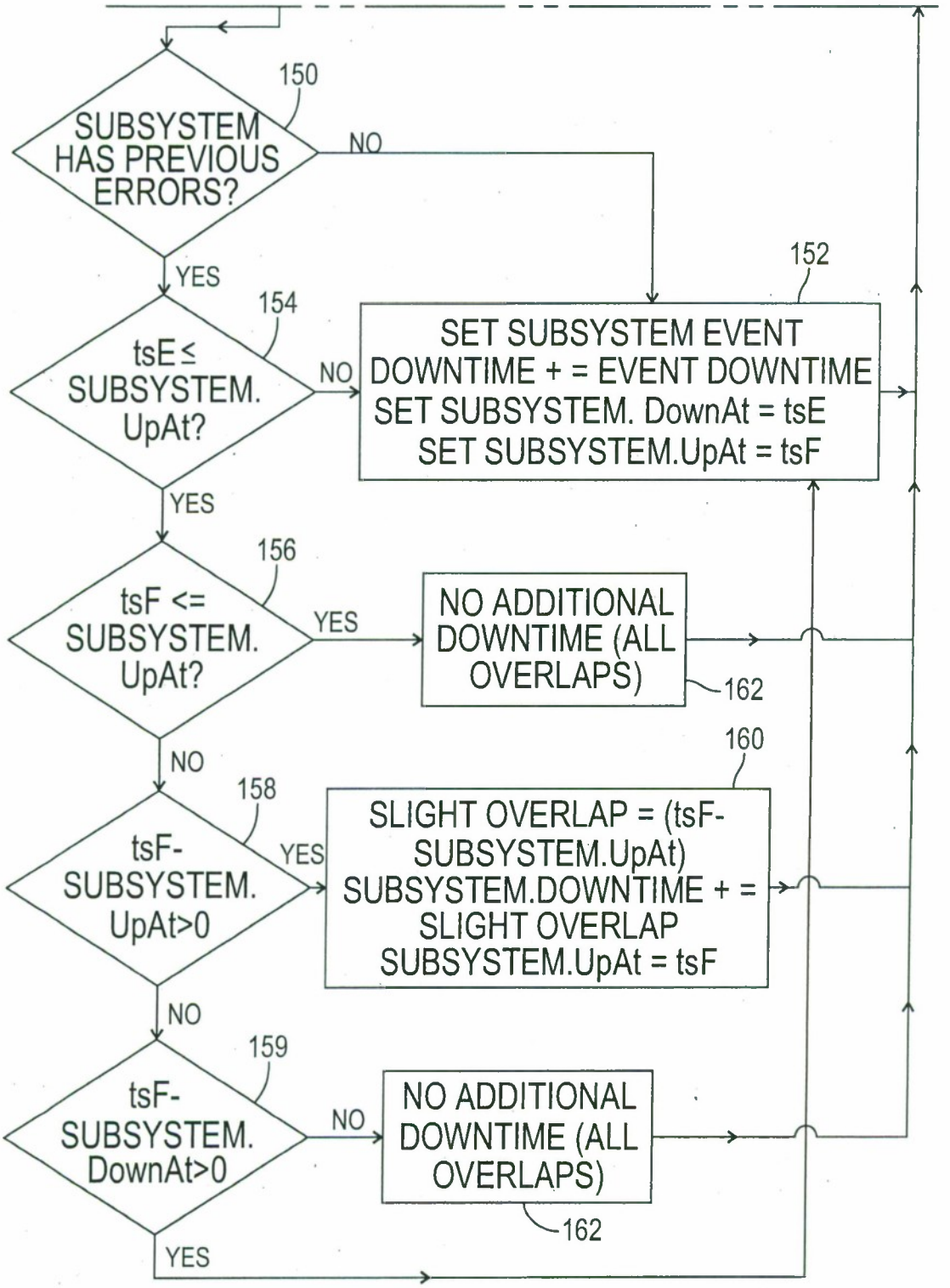
150
SUBSYSTEM HAS PREVIOUS ERRORS?
NO
YES
152
154
tsE ≤ SUBSYSTEM. UpAt?
NO
SET SUBSYSTEM EVENT DOWNTIME + = EVENT DOWNTIME
SET SUBSYSTEM. DownAt = tsE
SET SUBSYSTEM.UpAt = tsF
YES
156
tsF <= SUBSYSTEM. UpAt?
YES
NO ADDITIONAL DOWNTIME (ALL OVERLAPS)
162
NO
160
158
tsF- SUBSYSTEM. UpAt>0
YES
SLIGHT OVERLAP = (tsF- SUBSYSTEM.UpAt)
SUBSYSTEM.DOWNTIME + = SLIGHT OVERLAP
SUBSYSTEM.UpAt = tsF
NO
159
tsF- SUBSYSTEM. DownAt>0
NO
NO ADDITIONAL DOWNTIME (ALL OVERLAPS)
162
YES

FIG. 4C